Probabilistic Deception and Attacker Experimentation

Sharif Hassan and Ratan Guha  
Department of Computer Science  
University of Central Florida  
12760 Pegasus Drive  
Orlando, FL 32816, USA  
sharifhassan@knights.ucf.edu and guha@cs.ucf.edu  
Contact Author: Sharif Hassan  
SAM 2017 – Computer Security - Honeypot

Abstract—Cyber defensive deceptions are designed mostly for intentional attacks by cyber adversaries. However, they can also be susceptible to attacks within the deception unknown to the defender. Their successfulness in either case can vary based on their skill level and the overall landscape the deception resides. In this paper we characterize the attacker relationship with deployed deceptions. We define elements used to construct a decision tree that include the attackers tactics, techniques, and procedures, their skill level, the attack vectors within a deception, and indicators to detect. The tree describes the interactions between attackers and elements based on presumed conditions. Probability models based on the decision tree are presented as a foundation to developing equations that can be used to specifically calculate the success of an attacker and element interaction. We then validate our experience using a constructed simulator with two outlined scenarios each representing one side of an extreme in attacker skill level and landscapes. The results demonstrate an association where the average skill level changes the success of a deployed deception and its associated vulnerabilities, and indicators used to detect.

Index Terms— Cyber Defensive Deception, CND, Decision Tree, Probability Model, State Machine model

I. INTRODUCTION

Deception is an instrument that both the adversary and cyber defender capitalize. Bell and Whaley proposed two simple concepts of dissimulation, hiding the truth and simulate, showing the false [x]. Following this simple concept, the adversary may leverage deception to trick a victim into performing a “bad” action or perhaps to hide malicious code. For the defender deception can help confuse the attacker in thinking they are attacking the “real” system/data or as means to lure them to observe their tactics, techniques, and procedures (TTPs). Honeypots have a variety of configurations and many are automated applications and systems custom developed to meet a specific requirement. Some are larger projects while others unique but many share their honeypots purpose and code in online repositories. Code access allows for the discovery of potential vulnerabilities and the attacker becoming aware of the deception. This could lead to compromise attempts beyond the bounds of the deception against an unforeseen vulnerability [12]. Attackers vary in skill level and in a real-life case study researchers observed that the number of attacks is correlated to the complexity of the application being hosted on honeypots. As the deception was more sophisticated, the more attacker activity was observed [13].

Many models have been proposed to outline the progression and stages involved with deception in cyber defense. One such model is refer to as the Deception Incorporation model and is linear by design. The model starts with deception planning where the storyline and necessary components of the deception are developed. Then moves to implementing/integrating where the deception is visible to the attacker. The final stage is monitoring/evaluating the attacker activity with the final ending result being believed, suspected, or disbelieved [2, 10]. Another model, the Basic Deception-Process, was built for computer network defense (CND) operations and has a similar flow as in the Incorporated model except it allow for a decision gate for either continued deployment or to go back for further planning [9]. A third model, distinct from the other two, is the Spiral Cyber Denial and Deception lifecycle management process. This model is cyclical and allows for the deception to iterate through the planning, implementation, deploy/execute, and post-deployment analysis stages until deemed ready for production deployment [11].

In [1] the three models were characterized in an intuitive abstract model into three main states. The first being the ready state consisting of Plan & Deploy, the second the production state focused on Monitoring, and the third the determination state concentrated on Analysis. Seen in figure 1, the abstract model was the foundation to create a state machine model in [1] studying the effects to the system state as a deception is planned, deployed and monitored. Equation 1 taken from the state machine model starts at S0 where the elements that capture the potential change in state are α representing operating system files, β system processes, and μ network traffic and connections. A change to a system is not limited to just these three elements. Any number of elements can also be monitored to understand if a modification has occurred. If a particular event which caused a system state change is not captured under the elements being observed then the event goes unnoticed and the state of the system is not understood. As the planning on the system and eventual deployment of the deception occurs, the state changes to Sn. Equation 2 also taken from the state model denotes the attacker interacting with the deployed deception Dφ where φ is the attack type and λ the attack rate. As attacks continue
against the deception, TTPs are de-conflicted allowing for the deception to transition to the Determine state in which the deception ends or is recycled.

\[
S_0 = \langle \alpha_0, \beta_0, \mu_0 \rangle
\]  

(1)

\[
A[\phi, \lambda](\mathbf{D}_0) = D_k
\]  

(2)

In this paper we keep an abstract level to observe the inherent relationship between the attacker and deceptions. To study how effective deceptions are to different types of attackers and how varying landscapes can influence the successfulness of a deployed deception. We define elements used to construct a decision tree that include the attackers, deceptions, vulnerabilities, and indicators. We develop probability models based on the decision tree used to calculate the success of an attacker and element interaction. We validate our experience using a constructed simulator with two outlined scenarios each representing one side of an extreme in attacker skill levels and landscapes.

II. CHARACTERIZING THE Attacker

A. TTPs and Attacker Skill level

TTPs can be really anything the outline a particular action, behavior, or event executed by the attacker [3]. An example TTP may be a common encoding technique, a known exploit against a known vulnerability or sending an phishing email. TTPs vary in sophistication and ultimately dictates the skill level of a particular attacker. The skill level thus determines how successful any given vulnerability can be exploited. The capabilities and associated TTPs of an attacker can be grouped based their overall bias [2, 8]. Some bias lead to the conclusion of novice, “script kiddie” attackers while others are indicative of nation state organized military driven attackers [3]. The attacker bias helps frame the range of skill levels that a particular system may face and can ultimately dictate higher chances of a system compromise. Looking at equation 2 taken from the state diagram in figure 1, \( \phi \) is the type of attacker and \( \lambda \) is the rate of attack. Notionally these elements can be broadened and generalized under \( \beta \) as \( \phi \) being the element defining the attacker bias or skill level and \( \lambda \) being the element defining the accessibility by deception to allow the attack to occur. There also can exist more than two defined elements.

B. Attack Vectors

As the attacker interfaces with the deception, the attacker can attempt any potential attack vector that gains unintended access. This access can vary and may be serve a purpose such as a web application or be unknown as a zero day vulnerability. In the case of a deployed deception, the known vectors are managed and isolated to a particular object enticing an attacker to perform a nefarious act or to simply deceive, or preventing a certain action. This access can either be generalized as a vulnerability as it may be known (disclosed) or custom and specific, perhaps mimicking another known system. Ultimately this purposeful vector allows an attacker the ability to access the deception and hypothetically all other subsequent unintended attack vectors inside the deception. Vulnerabilities may be simple or complex requiring the right conditions and skill level to accomplish. Totality of this wide spectrum is the idea behind there being any number of possibilities in gaining access thus grouping them in one definition of commonality and intention.

C. Intelligence and Indicators

The intelligence, indicators, and threat data are the magnet looking for iron shavings in a pile of sand. As TTPs are conducted on the deception, events are occurring which can be previously known to the defender or similar in nature and thus can be characterized amongst all the other activity that is transpiring. This identified data can be then used to alert on the attackers presence. Intelligence, if you have the ability to possess it, is a key driver to successfully uncovering the adversary. In any situation, deception deployed or not, being cognizant of what the adversary performs is critical to understanding the security posture. The security posture as known or expected may indeed not be the posture as thought.
Intelligence driven defensive and synthesizing threat data is an approach that heavily relies on correlating the known intelligence to activity conducted by the adversary [4].

The decision tree represents several sets comprised of the elements attacker, deception, discovered vulnerability, and intelligence detection. Each element set may have different levels of complexity with any number at a particular level. Elements can change in size by adding/removing levels or increasing/decreasing the quantity at any level. For example there could be two attackers at skill level 1 and two at skill level 2. Later the landscape can change and there is now four at level 2 and three at a new level 3.

The first set contains the various attacker skill levels. Using \( \{A_1, A_2, \ldots, A_n\} \) where \( x \geq 0 \) and represents the number of attackers at a particular skill level, \( A_t \) is a single given attacker where \( 1 \leq t \leq n \) representing the skill level where novice begins at 1 and increases until the highest-level \( n \). It can be assumed that a particular attacker at skill level \( t \) has all the TTPs at that level or below. The second, third and fourth sets are deceptions, vulnerabilities, and indicators which is generalized as simply an element \( E \). Following the same approach using in the attacker set, elements follow \( \{E_1, E_2, \ldots, E_n\} \) where \( x \geq 0 \) and is the number of elements at each level. \( E_t \) is a single given element where \( 1 \leq t \leq n \) and level \( 1 \) being minimal to \( n \) being particularly difficult.

There also exist conditions in table 1 between the elements that constitute how they operate. Following these conditions allows the tree to process what decisions to make based on the provided criteria, in most cases the attacker skill level compared to the level of the other elements. In addition we infer the following assumptions: (1) It is conceivable an attacker "plays" out the deceptions storyline and pretends to be deceived while trying to find other attack vectors. For simplicity, we assume if an attacker is deceived, then they do not look for vulnerabilities. (2) If an attacker is suspicious of a deception, then they must be higher in level than a particular vulnerability in order to successfully exploit versus equal or greater than if the deception failed.
Table I – Decision Tree Conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>V</th>
<th>D</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_i &gt; E_i</td>
<td>Successful attack if D = suspicious or failed</td>
<td>Deception Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>A_i ≥ E_i</td>
<td>Successful attack if D = failed</td>
<td>Deception Suspicious</td>
<td>Match</td>
</tr>
<tr>
<td>A_i &lt; E_i</td>
<td>Failed attack if D = suspicious or failed</td>
<td>Deception Successful</td>
<td>Match</td>
</tr>
<tr>
<td>A_i ≤ E_i</td>
<td>Failed attack if D = suspicious</td>
<td>----</td>
<td>Match</td>
</tr>
</tbody>
</table>

IV. Attacker Deception Probability Models

Using the decision tree and the conditions outlined above, defined probability equations are modeled on the interactions between the deceptions, vulnerabilities, and indicators. The probabilities are divided into four main models that focus on any attacker to any element, select attacker to any element, select attacker to select element, and any attacker to select element. Select meaning all possible items of a particular element at a specific level. So if A_i was selected, then the set is to include all of xA_i. The compliment is all other items at all other levels. The term any is used as the enumeration of all items for a particular element at all levels. This leads to the spectrum where any to any is the theoretical and select to select is empirical and the select to any and any to select is an amalgam between the two.

A. Deriving probabilistic equations

Equation 1 represents the attacker where A can equal either a selected attacker t over all possible attackers na or can be any attacker I that exists between 1 and n which is also over all possible attackers na. Expanding a_t, is the sum of all attackers at a selected attacker skill level. Expanding a_i is the sum of all attackers at all attacker skill levels. Equation 2 represents the various elements being deceptions, vulnerabilities, and indicators. Equally E can either be a selected element t over all possible elements ne or can be any element i between 1 and n over all possible elements ne. In both equations n is the highest level for that element and x is the total count at a particular level.

\[ A = \left\{ \frac{a_t}{na}, \frac{a_i}{na} \right\} \]

\[ a_t = \sum_{1}^{x} A_t, a_i = \sum_{1}^{x} A_i \]

\[ A \leq \left\{ \frac{e_t}{ne} \right\} \]

\[ n = \sum_{1}^{x} \sum_{1}^{x} a_i \]

\[ E = \left\{ \frac{e_t}{ne} \right\} \]

\[ n = \sum_{1}^{x} \sum_{1}^{x} e_i \]

Each derived probabilistic equation has the basic foundation of equation 3 where the attacker is computed with some other element. Each follow the selected or any notion meaning A could be a selected attacker over all possible attackers and E could be a selected deception over all possible deceptions. Based on the circumstances, the conditions are applied as outlined in the decision tree. For example in the case where the equation is developed to determine the probability an attacker is successfully deceived, the condition needs to be considered where the deception level has to be greater than the attacker skill level.

\[ A \times E \]

Observing how the probabilistic equations are constructed, the probability of a successful deception for any attacker to any deception for all attackers that are greater than the deception is seen in equation 4 where d_i is substituted for e_i. The outer summation, m is equal to the highest level of any element involved in the particular computation. Meaning if there were 3 attacker skill levels and 2 deception difficulty levels, then m = 3.

\[ \sum_{i=1}^{m} \left( \frac{a_t}{na} \times \frac{d_i}{na} \right) \forall \ a_i > d_i \]

It is easy to change an equation to fit another situation or other elements by simply making the right modifications. For example changing equation 4 from any attacker to any deception (AAAD) into selected attacker to any deception (SADD) a_i is simply changed to a_t. Likewise to change from a SADD to a SASD, e_i is changed to e_t. Both equations can be seen in 5 and 6. Equations 4-6 can also be used for calculating the probability of an attacker successfully compromising a vulnerability if the deception is suspicious with the only change being V versus D substituted in for E.

\[ \sum_{i=1}^{m} \left( \frac{a_t}{na} \times \frac{d_i}{na} \right) \forall \ a_t > d_i \]

The conditions can also be the focal point on where the equation deviates slightly for one another. Comparing equation 5 SSSD to equation 7 SASV, the only variance is in the equality within the condition. Equation 8 for SASI has the
same equality operator as equation 7 but the operands are in opposite sides.

$$\sum_{i=1}^{m} \left( \frac{a_t}{n_a} \times \frac{v_t}{n_a} \right) \forall \ a_t \geq v_t$$  \hspace{1cm} (7)

$$\sum_{i=1}^{m} \left( \frac{a_t}{n_a} \times \frac{i_t}{n_a} \right) \forall \ i_t \geq a_t$$  \hspace{1cm} (8)

These example equations are a subset of what could be created to fit the right condition describing the interaction among various elements. The equations thus far concentrated on the success of a particular element but the compliment can just as easily be derived. Furthermore these equations can be concatenated and substituted to form other probabilistic models such as the best case or worst case scenarios namely the attacker is not deceived, discovers and exploits a vulnerability, and is not detected by an indicator.

B. Examples

Example probability calculations are performed using the landscape in table II. In this case m equals 3 since all have the same highest level of 3.

<table>
<thead>
<tr>
<th>Attacker</th>
<th>Vulnerability</th>
<th>Deception</th>
<th>Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>V1</td>
<td>D1</td>
<td>I1</td>
</tr>
<tr>
<td>A2</td>
<td>V2</td>
<td>D2</td>
<td>I2</td>
</tr>
<tr>
<td>A3</td>
<td>V3</td>
<td>D3</td>
<td>I3</td>
</tr>
</tbody>
</table>

The probability of any given attacker successfully exploiting any vulnerability is calculated below using a similar equation seen in 4 except the equality is greater than equal to vs. just greater than due to the condition.

$$\left( \frac{3}{6} \times \frac{3}{6} \right) + \left( \frac{2}{6} \times \frac{5}{6} \right) + \left( \frac{1}{6} \times \frac{6}{6} \right) = .70$$

Where the complement of any given attacker not successfully exploiting a system is calculated below.

$$\left( \frac{3}{6} \times \frac{3}{6} \right) + \left( \frac{2}{6} \times \frac{1}{6} \right) + \left( \frac{1}{6} \times \frac{0}{6} \right) = .30$$

The probability of a selected attacker, say A2 successfully exploiting a selected vulnerability, say V1 using equation 7 is calculated below.

$$\left( \frac{2}{6} \times \frac{3}{6} \right) = .1666$$

As an example of combining the equations together, the worst-case of any attacker being successfully exploiting any vulnerability, not being deceived by any deception, and not matching any indicator is calculated.

$$\text{Probability} = 0.70 \times 0.388 \times 0.638 = 0.137 \text{ or } 13.7\%$$

V. Probabilistic Simulation

A simulator using python and NumPy was developed to replicate the interactions between elements following the outlined conditions in the decision tree [5]. The intention is to demonstrate and observe the effects of a changing landscapes on the overall success probability. At load time, the number of levels and quantity at each level are configured for all the elements. Using this preliminary landscape, the simulation executes a user-defined number of cycles each having a user defined number of iterations. Within each iteration, probabilities of all elements are calculated for all model types, AAAD, AASE, SAAE, and SASE. Upon the end of the cycle, all iteration calculations are summed, averaged and recorded. At the completion of all cycles, the results are then summed and averaged to produce the final probability summations.

A. Simulation One – Dark Military

This simulation will observe nation state attackers being introduced into the landscape over a period of time with subsequent indicator data being added to detect the new attacker TTPs. This simulation has 200 cycles each with 10000 iterations. On cycle 75-124, fifty new level four attackers are introduced representing nation state capabilities with zero day TTPs. On cycle 125-174 fifty new level four indicators are introduced to detect the new attackers.

Table III displays the probability results and element levels as the simulation is executed. The initial landscape is seen in...
row one of the table with the end landscape in the last row, cycle 200. Studying the results, the addition of advanced attackers reduced the success probability for both deceptions and indicators while increased for vulnerabilities. Observing one of the models AAAD, the success probability decreased almost 23% from cycle 1 to cycle 200 while AAAV increased just over 20%. As new level four indicators were introduced in cycle 125, the resulting AAAI success probability ended slightly lower at .6916 than its original starting point of .7751 Figures 4, 5, and 6 illustrate the results of Table III.

Combining the three elements together to make up the worst case, figure 7 shows the merged probability of a failed deception, successful vulnerability exploit, and failed indicator detection. Before the advanced attackers were introduced, the probability was below 10%. Inherently a steady incline was observed as the advanced attackers were introduced into the landscape. As the eventual indicators are applied, the probability begins to cutback tailing off close to 20%. The difference between the starting 10% and ending 20% is due to now having fifty level four attackers in the mix.

**B. Simulation Two – Graduating Script Kiddies**

This simulation focused on the exploding numbers of novice script kiddie to mid tier level attackers. As cyber security is being taught in schools and there is an ever increasing demand for cyber talent, more are attempting to become “hackers” typically staying within the realm of executing tools that are coded by experienced security professionals or advanced attackers and some eventually prompting to higher level TTPs. [6]. The goal of this scenario is to see how the continued numbers of low to mid level hackers being added to the landscape changes the overall
deception levels, the chances of exploitation, and the ability to detect them. As in the Dark Military scenario there are 200 cycles each with 10000 iterations. At the start of each new cycle, a new level 1 attacker is added.

Table IV displays the probability results and element levels as the simulation is executed. The initial landscape is depicted in the top row and ending landscape in the bottom row. The continued introduction of novice level one attackers increased the AAAD success probability from .4152 in cycle 1 to .6448 in cycle 200, a 22% gain. This is a direct result of level one attackers being easily deceived by higher-level deployed deceptions. Indicators also exhibited a higher success probability. Vulnerabilities however decreased as more novice attackers took advantage of easier level one vulnerabilities.

Table IV - Element levels and Probabilities results for Graduating Script Kiddies simulation execution

<table>
<thead>
<tr>
<th>Cycle</th>
<th>A Lvl</th>
<th>D Lvl</th>
<th>V Lvl</th>
<th>I Lvl</th>
<th>AAAD</th>
<th>SAAD</th>
<th>AASD</th>
<th>SAD</th>
<th>AAV</th>
<th>SAAV</th>
<th>AASV</th>
<th>SAV</th>
<th>SASD</th>
<th>SAD SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111</td>
<td>81</td>
<td>61</td>
<td>81</td>
<td>.4152</td>
<td>.2290</td>
<td>.1902</td>
<td>.1073</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>72</td>
<td>102</td>
<td>72</td>
<td>.7870</td>
<td>.3627</td>
<td>.2595</td>
<td>.1278</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-199</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>.6345</td>
<td>.2316</td>
<td>.2669</td>
<td>.0999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>210</td>
<td>81</td>
<td>61</td>
<td>81</td>
<td>.6448</td>
<td>.6193</td>
<td>.3033</td>
<td>.2914</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>72</td>
<td>102</td>
<td>72</td>
<td>.9104</td>
<td>.9293</td>
<td>.3309</td>
<td>.3139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>.4610</td>
<td>.4135</td>
<td>.2031</td>
<td>.1841</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Jumping to the worst case, figure 8 reveals a different outcome than the Dark Military scenario. Initially the probability for the model AASE was close to 75% while a majority of the other configuration models started near 30%. As novice attackers continued to flood the landscape, the probability continues to decrease. This is a direct result of script kiddie attackers not having the necessary aptitude to successful execute the worst-case scenario.

Fig. 8. Worst Case probabilities for fail deceptions, successful vulnerability exploits, and failed indicator deception in Graduating Script Kiddie scenario showing from top to bottom AAAD, SAAD, AASD, and SASD.

VI. CONCLUSION

We showed the relationship between attackers and other elements at an abstract level using a decision tree and proposed probability models. A simulation was used with varying attacker and deception levels being introduced into the landscape. Two scenarios were created with the first introducing advanced attackers and indicators and the second focused on flooding the landscape with greenhorn attackers. Looking at the results for both scenarios proved the varying skills do have a profound impact on the ability for an attacker to be successfully deceived, an attacker to successfully exploit an unidentified vulnerability, and whether the attacker is successfully detected.

REFERENCES
